Claims

1.- A computer implemented method for classifying objects into a set of global objects, containing objects that can be reached by more than one thread, and a set of local objects, containing objects that can only be reached by one thread, when executing multi-threaded programs, whereby the classifying is done dynamically by observing modifications to references to objects by operations performed in the computer system.

2.- The method of claim 1, wherein each object is provided with an instrumentation data structure to enable observation of modifications to references to objects.

3.- The method of claim 2, wherein the instrumentation data structure comprises at least a thread identification tag for identifying whether an object can be reached by only one thread or by more than one thread.

4.- The method of claim 1, further comprising the step of:
recording in a memory concurrency state transition information of global objects.

5.- The method of claim 1, further comprising a step of selectably performing garbage collection only on the set of local objects.

6.- A computer implemented method for detecting inconsistent dynamic concurrency state transitions in the execution of multi-threaded programs amenable to object reachability analysis, the method comprising the steps of:
executing multiple threads on a computer;
at least periodically during execution of the threads classifying instantiated objects into a set of global objects (503; 1508), containing objects that can be reached by more than one thread, and sets of local objects (504; 1505, 1506, 1507), containing objects that can only be reached by one thread, and
recording in a memory concurrency state transition information of global objects.

7.- The method according to claim 6, further comprising the step of determining occurrence of data races between two or more threads.

8.- The method according to claim 6, wherein the set of global objects (503; 1508) is periodically analysed so as to remove objects from the global set which are only reachable by one thread.

9.- Method according to claim 8, wherein the removal of objects of the global set which are only reachable by one thread takes place after a garbage collection step.

10.- Method according to claim 6, wherein each object created during execution of the program is provided with an instrumentation data structure (404) to enable race detection.

11.- Method according to claim 10, wherein the instrumentation data structure (404) comprises

- a first address field (405) for containing the address of a first data structure (409; 1401) when the object is locked for the first time,
- a second address field (406) for containing the address of a second data structure (410; 1301) if the object is of type thread or a subtype thereof,
- a thread identification field (407) for recording whether the object belongs to the global set or to a local set,
- a third address field (408) for containing the address of a third data structure (411; 1601) when the object belongs to the global set.

12.- Method according to claim 11, wherein the first data structure (409; 1401) comprises a vector clock field for taking into account the fact that threads are created and destroyed dynamically.

13.- Method according to claim 11, wherein the second data structure (410; 1301) comprises a vector clock field (1302) taking into account the fact that threads are created and destroyed dynamically, and indicating the current vector clock for the currently executing event on the thread, and a thread identification number (1303).

14.- Method according to claim 11, wherein the third data structure (411; 1601) comprises a counter field (1602) for containing the number of member variables present in the object, and a fourth address field (1603) for containing the address of a fourth data structure (1604).

15.- Method according to claim 14, wherein the fourth data structure (1604) is an array of a length given in the counter field that contains addresses of fifth data structures (1606).

16.- Method according to claim 15, wherein the fifth data structures (1606) comprise

- a field description field (1607) describing the member variable that is being observed,
- a lock field (1608) for obtaining exclusive access to the fifth data structure (1606)
- a read list address field (1609) for containing the address of a doubly linked list of read information structures (1611) used to record information on relevant read operations
- a write list address field (1610) for containing the address of a write information data structure (1618) used to record information on relevant write operations.

17.- Method according to claim 16, wherein the read information structure (1611) describes the most recent read operation performed for each separate thread on the corresponding member variable.

18.- Method according to claim 16, wherein the write information data structure (1618) describes the most recent write operation performed on the corresponding member variable.

19.- A computer implemented method for determining the order of events in the presence of a dynamically changing number of threads of a computer program executable on a computer, wherein a clock data structure (601) is maintained in memory from which clock data structure the occurrence of two events in parallel during execution of the threads can be determined, the dimension of the clock data structure (601) being determined dynamically dependent upon the number of threads created and destroyed during execution of the program.

20.- Method according to claim 19, wherein the clock data structure (601) comprises a lock (602) to be taken by a thread if exclusive access to the clock is required, and the address (603) of a local clock data structure (604).

21.- Method according to claim 20, wherein the local clock data structure (604) comprises

- a lock field (605) to be locked by a thread for obtaining exclusive access to the local clock data structure (604),
- a count field (606) indicating the number of different vector clocks that use the local clock data structure (604),
- an array of values (607) containing the values of he local clock data structure (604),

- a next field (608) and a previous field (609) to link the local clock data structure (604) in a doubly linked circular list (610).

22.- Use of the method according to claim 1, whereby the method is implemented in association with a virtual machine.

23.- The use according to claim 1, wherein the method is implemented in an interpreter.

24.- Use of the method of claim 1, whereby the method is implemented in a compiler.

25.- Use of the method according to claim 1, whereby the method is implemented in hardware.

26.- Use of the method according to claim 1, whereby the method is implemented in a garbage collector.

27.- A computer readable data carrier comprising a computer executable computer programming product for executing the method of claim 1 on a computer.